REMARKS

In the Office Action of July 30, 2003 ("Office Action"), claims 1-23 were rejected. The specification and claims are amended as shown above. Reconsideration of the application in view of such amendments and the following remarks is respectfully requested.

Interview

Applicant thanks the Examiner for the courtesy extended to Applicant's representative in granting an interview to discuss the present application on October 7, 2003. The rejection with respect to use of the word Java was discussed, and Applicant's representative indicated that an amendment would be submitted with regard to the claims using such language.

The rejection of claim 1 was discussed. Applicant explained that Tock is an offline class loader, and that Tock fails to anticipate what is claimed in claim 1. Applicant explained that although Tock teaches some activities that are performed during runtime, Tock fails to teach that the activities listed in the Office Action are all performed during runtime, and therefore, the Office Action fails to make a sufficient showing that the claims of the present application are anticipated by Tock.

The Examiner pointed to certain portions of Tock involving runtime activities. Applicant's representative indicated that the Office Action is still incorrect in stating that Tock teaches that the particular sequence of functions are to take place during runtime, and rather, that Tock teaches other functions that are to take place during runtime.

It was agreed that Applicant would submit a response to the Office Action explaining why the rejection should be removed and including the above-mentioned amendment of claims.

The Specification

The Office Action indicates that the application number of the incorporated patent application should be updated. Applicant has amended the specification as shown above and respectfully requests review and approval of the change. It is believed that the concern raised in the Office Action has been overcome.

The Examiner states the use of the trademark Java™ in the application should be capitalized and accompanied by the generic terminology throughout the application. It is believed that the Office Action did not require any change to the specification with respect to "Java," but is merely making a cautionary comment. Thus, no amendment has been made to the specification in this regard. An amendment has been made to certain claims that use the word "Java," as discussed below.

Amendment to Correct Informality

Claims 1, 9 and 16 have been amended to add the word "and" to correct an informality so as to correctly render the phrase "if and only if." Review and approval are respectfully requested.

Rejection of Claims 3, 5, 10, and 17 under 35 USC § 112, Second Paragraph

Claims 3, 5, 10, and 17 were rejected for containing the trademark/trade name Java™. Applicant respectfully traverses the rejection. While it is not necessarily believed that the use of the word Java renders such claims unpatentable, in order to expedite the application to allowance, such claims have been amended to eliminate such language and provide other language. Review and approval are respectfully requested.

Rejection of Claims 14, 22 and 23 under 35 USC § 112, Second Paragraph

Claims 14, 22 and 23 were rejected as being indefinite.

With respect to claims 14 and 22, it is believed that such claims made reference to the incorrect parent claims as a result of a typographical error. A correction has been made. It is believed that this correction also corrects the concern regarding antecedent basis, since the "larger indexes" language is recited in the respective parent claims. Accordingly, it is believed that the rejection has been overcome and removal of the rejection is respectfully requested.

With respect to claim 23, the language "first and second" has been added. It is believed that such correction overcomes the rejection, being consistent with the way the Office Action indicates that the Examiner interpreted the claim. Removal of the rejection is respectfully requested.

Rejection of Claims 1-23 under 35 USC § 102

Claims 1-23 were rejected as being anticipated by U.S. Patent No. 5,815,718 (Tock). The Applicant respectfully traverses the rejection.

The Office Action indicates that Tock teaches a method substantially as claimed for loading Java™ classes as needed at run-time (See Office Action, p. 5).

It is believed that the characterization of Tock in the Office Action is not correct. In particular, Tock does not teach:

> for data structures in a set of data structures, as unloaded data structures are needed during runtime,
>
> receiving a data structure from a first memory, the data structure including one or more sets of instructions and one or more constants;
>
> storing instructions from the data structure in a first portion of a second memory, the second memory comprising RAM;

storing constants from the data structure in a second portion of the second memory if and only if the respective constant has not been stored in the second portion of the second memory,

modifying indexes in instructions that reference the constants to correspond to the respective locations of the constants in the second portion of the second memory, and

reading and executing at least some instructions from the data structure from the RAM.

In particular, Tock teaches an "offline class loader," and thus teaches away from the approach claimed in claim 1 which involves activities performed for data structures in a set of data structures, as unloaded data structures are needed during runtime.

The Office Action appears to be incorrectly mixing and matching disconnected portions of Tock in a way not disclosed or suggested by the reference. For example, the Office Action points to a disclosure of "run-time" in the abstract of Tock, and points to other items apparently disclosed in Tock, and then makes the incorrect logical leap that all such other activities occur in run time. This interpretation is incorrect.

For example, the Office Action points to col. 1, li. 41-46 of Tock. This portion of Tock indicates "... this disclosure pertains to an offline class loader that is used to produce an executable module whose classes are preloaded into memory without requiring runtime dynamic loading. The executable module, nevertheless, contains a class structure that is tailored for runtime dynamic loading. ..." It is true that this section of Tock mentions "runtime dynamic loading." However, this mention of runtime dynamic loading does not teach or suggest that the particular activities are performed as unloaded data structures are needed during runtime. Rather, Tock teaches a number of activities that are performed off-line, and other activities that are performed during runtime. Thus, Tock fails to teach or suggest the particular approach claimed in claim 1.

The Office Action's attempt to make out a rejection of claim 1 is improper in view of M.P.E.P. 2131, which states, in part, "The elements must be arranged as required by the claim... ." See M.P.E.P. 2131, paragraph 2, entitled "To Anticipate a Claim, the Reference Must Teach Every Element of the Claim." The Office Action is improperly rearranging elements of Tock, in violation of this requirement of M.P.E.P. 2131, to try to make out an anticipation rejection of claim 1. The elements cited in the Office Action regarding runtime do not occur in the order stated in the Office Action. Thus, for this additional reason, the rejection is improper in view of M.P.E.P. 2131.

Further, since Tock teaches an "offline class loader," Tock teaches away from the approach claimed in claim 1 which involves activities performed for data structures in a set of data structures, as unloaded data structures are needed during runtime. The Office Action is confusing activities that take place off-line as taught by Tock with activities that take place during runtime.

Thus, for the reasons discussed above, it is believed that the rejection of claim 1 has been overcome and should be removed.

Claims 2-8 depend from claim 1. Since the rejection of claim 1 has been overcome, the rejection of these dependent claims, which depend from claim 1, has also been overcome. Review and approval of such claims are respectfully requested.

With respect to claim 9, the Office Action points to the rejection of claims 1, 2 and 4. Since the rejection of claim 1 has been overcome for the reasons set forth above, it is believed that the rejection of claim 9 should also be removed. Further, the rejection of claims 10-15, which depend from claim 9 should also be removed because the rejection of claim 9 has been overcome.

The Office Action indicates that claim 16 is rejected for the reasons stated above in the Office Action. Since the rejection of claim 1 has been overcome as set forth above, the rejection of claim 16 should also be removed. Claim 16 has also been amended, and review and approval of such amendment is respectfully requested. Claims 17-23 depend from claim 16, and are thus patentable for at least the reasons as to claim 16. Therefore, the removal of the rejection of such claims is respectfully requested.

Comments to Interview Summary

The Examiner's Interview Summary points to certain aspects of Tock that relate to runtime activity and the use of RAM. However, even in view of the portions of Tock cited by the Examiner, Tock still fails to anticipate claim 1. The Examiner seems to imply that since Tock teaches that some loading or other activities may take place during execution, that claim 1 is anticipated. Applicant does not agree with this interpretation of Tock. In particular, the Office Action and the portions of Tock cited by the Examiner in the Interview Summary fail to show how Tock would teach,

> for data structures in a set of data structures, <u>as unloaded data structures are needed during runtime</u>,
>
> receiving a data structure from a first memory, the data structure including one or more sets of instructions and one or more constants;
>
> storing instructions from the data structure in a first portion of a second memory, the second memory comprising RAM;
>
> storing constants from the data structure in a second portion of the second memory if and only if the respective constant has not been stored in the second portion of the second memory,
>
> modifying indexes in instructions that reference the constants to correspond to the respective locations of the constants in the second portion of the second memory, and
>
> reading and executing at least some instructions from the data structure from the RAM.

Rather, the Examiner has pointed to certain activities that take place during runtime. The Office Action and the Examiner's comments in the Interview Summary have not shown that all of the above steps take place "for data structures in a set of data structures, as unloaded data structures are needed during runtime."

For example, the Interview Summary points to col. 3, lines 14-18 of Tock. This portion of Tock states: "The second address space resides in a read/write memory device, such as a random access memory, and contains methods that require dynamic loading and data that is varied during execution." This portion of Tock is referring to the way that the offline class loader partitions a Java application, as described in the previous portion of this paragraph of Tock:

> The offline class loader partitions a Java application, such as the browser and the basic support classes, into at least two separate address spaces. A first address space resides in a read-only memory device and contains methods that do not require dynamic loading and data that remains constant. The second address space resides in a read/write memory device... .

Tock, column 3, lines 10-15. Thus, this is referring to the way that the offline class loader partitions memory. This is not a teaching that the claimed steps all take place during runtime.

The Interview Summary further indicates that "the dynamic data is stored in RAM." This assertion also fails to support an argument that Tock anticipates the invention. The Interview Summary points to column 5 lines 21-26 and column 1, lines 47-49 of Tock, and states that "which indicates that varying (runtime) data and method that contain unresolved references are provided for." Even if true, this fails to show how Tock would anticipate the invention. Column 5 lines 18-26 of Tock states:

> The goal of the offline class loader 132 is to determine which methods and variables associated with each class can be stored in a read-only memory and which must be stored in a random access memory device. Methods that invoke Java interfaces or utilize non-

> static instance variables need to reside in random access memory. This is because the bytecodes that implement interfaces are determined at runtime and non-static instance variables are altered for each instantiation of the associated class.

As can be seen from the underlined text above, Tock is teaching actions of the offline class loader. The result is information that applies to what methods and variables are stored in read-only memory and random access memory. However, Tock is not teaching the process of claim 1, with all its associated steps taking place for data structures in a set of data structures, as unloaded data structures are needed during runtime.

The other cited portions of Tock, such as column 1, lines 47-49 and 60-67 similarly fail to teach the process of claim 1, with all its associated steps taking place for data structures in a set of data structures, as unloaded data structures are needed during runtime. For example, Tock, at column 1, lines 60-67 indicates:

> A preloadable executable module of this fashion is advantageous in a distributed computer system having client computers with little or no secondary storage. Such client computers require applications to run entirely in random access memory which quickly turns into a limited resource. By utilizing the offline class loader to partition an application into two address spaces, the amount of RAM utilized by the preloadable module is minimized.

This portion of Tock is talking about the use of the output of the offline class loader in a computer system having RAM. However, this is a completely different approach than claim 1, which states:

> for data structures in a set of data structures, as unloaded data structures are needed during runtime,
>
> receiving a data structure from a first memory, the data structure including one or more sets of instructions and one or more constants;
>
> storing instructions from the data structure in a first portion of a second memory, the second memory comprising RAM;
>
> storing constants from the data structure in a second portion of the second memory if and only if the respective constant has not been stored in the second portion of the second memory,

modifying indexes in instructions that reference the constants to correspond to the respective locations of the constants in the second portion of the second memory; and

reading and executing at least some instructions from the data structure from the RAM.

In other words, Tock teaches an offline class loader, and certain activities that take place during runtime using the result of the offline class loader. Tock fails to teach the process of claim 1, with all its associated steps taking place for data structures in a set of data structures, as unloaded data structures are needed during runtime.

Further, Tock teaches at column 1, lines 41-49:

> In summary, this disclosure pertains to an offline class loader that is used to produce an executable module whose classes are preloaded into memory without requiring runtime dynamic loading. The executable module, nevertheless, contains a class structure that is tailored for runtime dynamic loading. Thus, the offline class loader modifies the existing class structures to accommodate static loading. However, the class structure allows for varying data and methods that contain unresolved references.

Thus, most of the teaching of Tock is directed to the offline class loader. As indicated in the above-cited text, the class loader is used to produce an executable module whose classes are preloaded into memory without requiring dynamic loading. However, the executable module, nevertheless, contains a class structure that is tailored for runtime dynamic loading. Thus, Tock is teaching some activity that takes place during runtime, namely that the class structure is tailored for runtime dynamic loading. However, this does not mean that Tock teaches, and in fact Tock fails to teach:

for data structures in a set of data structures, as unloaded data structures are needed during runtime,

receiving a data structure from a first memory, the data structure including one or more sets of instructions and one or more constants;

storing instructions from the data structure in a first portion of a second memory, the second memory comprising RAM;

storing constants from the data structure in a second portion of the second memory if and only if the respective constant has not been stored in the second portion of the second memory,

modifying indexes in instructions that reference the constants to correspond to the respective locations of the constants in the second portion of the second memory, and

reading and executing at least some instructions from the data structure from the RAM.

This portion of Tock makes clear that Tock teaches away from the approach claimed in claim 1.

Conclusion

For the reasons set forth above, Applicant believes that the present application is in condition for allowance, and respectfully request issuance of a timely Notice of Allowance. Should the Examiner have any questions regarding this application or response, the Examiner is invited to contact Applicant's representative at the number set forth below.

Respectfully submitted,

Wilson Sonsini Goodrich & Rosati

By _____

George A. Willman
Reg. No. 41,378
Tel.: (650) 320-4945
Customer No. 021971